بسمه تعالی

تمرین سری هشتم درس یادگیری عمیق

پوریا محمدی نسب

۴۰۰۷۲۲۱۳۸

فهرست

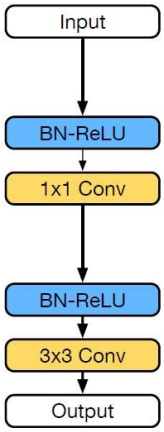
سوال ۱.

مقاله زیر با بهبود دادن ایده مقاله DenseNet معماری قدرتمندی برای دسته بندی تصاویر ارایه داده است. با مطالعه این مقاله، ایده های آن را توضیح دهید.(برای پاسخ دادن به این سوال مطالعه قسمت های ابتدایی مقاله و بررسی اجمالی معماری آن کافی است اما مطالعه کامل این مقاله کاربردی توصیه میشود.)

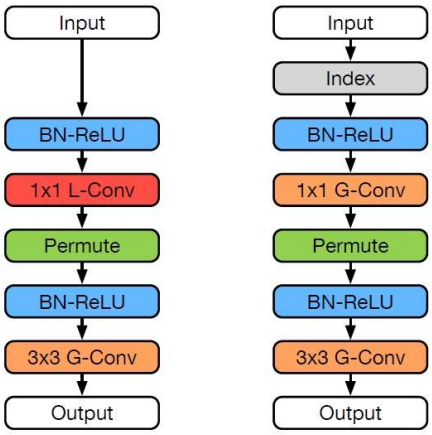
Huang, Gao & Liu, Shichen & van der Maaten, Laurens & Weinberger, Kilian." CondenseNet: An Efficient DenseNet Using Learned Group Convolutions."CVPR, 2018.

شبکه های کانولوشنی عمیق به طور فزاینده ای در دستگاه های تلفن همراه مورد استفاده قرار میگیرند که این دستگاه ها از منابع محاسباتی محدودی برخوردار هستند. در مقاله مورد بررسی یک معماری مدل جدید با نام ConenseNet معرفی میشود که ویژگی مهم این معماری کارایی بسیار خوب و بی سابقه ی آن است. ایده اصلی این مدل ترکیب dense معرفی شده در مدل denseNet با ماژول جدیدی به نام گروه کانولوشنی آموزش دیده(learned group convolution) است. کار ماژول Dense در این معماری استفاده دوباره از ویژگی ها است در حالی که learned group convolution اتصالات بین لایه ها را حذف میکند زیرا استفاده دوباره از بعضی ویژگی ها در معماری DenseNet اضافه است. در ادامه به صورت مفصل تری DenseNet و Group Convolution را مورد بررسی قرار میدهیم.

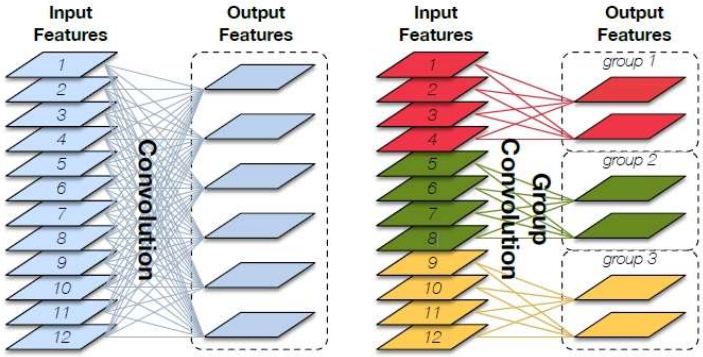
DenseNet: این مدل شامل چندین Dense block است که هر کدام از این واحدها به تنهایی چندین لایه در خود دارند. هر لایه در یک بلوک Dense، k تا فیچر تولید میکند که تعداد k به نرخ افزایش رشد شبکه بستگی دارد. ویژگی متمایز کننده DenseNet این است که ورودی هر لایه شامل اتصال تمام نقشه های ویژگی (feature map) تولید شده از لایه های قبلی در یک block است. معمولا در یک بلوک لایه اول با استفاده از فیلتر های 1*1 تعداد کانال های وردی را برای صرفه جویی در هزینه کاهش میدهد. شکل زیر نمایش دقیق تری از لایه های مورد استفاده در یک Dense block است.



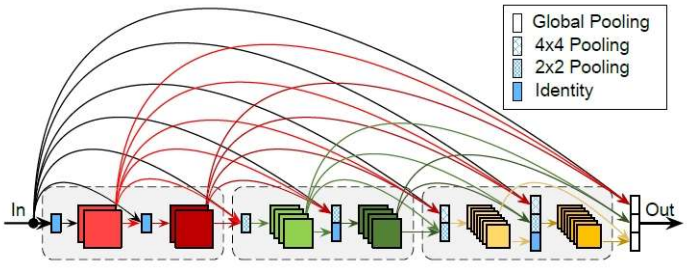
در مدل پیشنهادی تغییراتی در بلوک شکل بالا انجام میشود. نکته جالب در مورد مدل پیشنهادی این است که در زمان آموزش و آزمون از دو نوع بلوک متفاوت استفاده میکند. شکل زیر سمت چپ بلوک تغییرکرده Dense در زمان آموزش را نمایش میدهد و شکل سمت راست مدل پیشنهادی برای زمان آزمون است.



Group Convolution: این تکنیک برای اولین بار در معماری AlexNet استفاده شد و سپس به خاطر موفقیت و نتایج خوبی که داشت در مدل ResNext معروف شد. کانولوشن استاندارد با R نرون در ورودی و O نرون خروجی به صورت کاملا متصل کار میکند که هزینه محاسباتی آن R*O است. در طرف مقابل group convolution با گروه بندی کردن ویژگی های ورودی به G دسته که هیچ اشتراکی با هم ندارند باعث میشود تا هزینه محاسباتی به R*O)/G) برسد. شکل زیر ساختار کانولوشن استاندارد (سمت چپ) و کانولوشن گروهی (سمت راست) را نمایش میدهد.



در انتها مدل پیشنهاد شده با ترکیب دو مورد بالا و اعمال یک سری از تغییرات جزئی در Hyperparameter های DenseNet به نتایج بسیار بهتری نسبت به سایر مدل ها رسیده است. شکل زیر خلاصه ای از تغییرات جزئی است که شامل تغییر ابعاد feature map ها و همین طور نرخ رشد این مدل دو برابر مدل قبلی است.



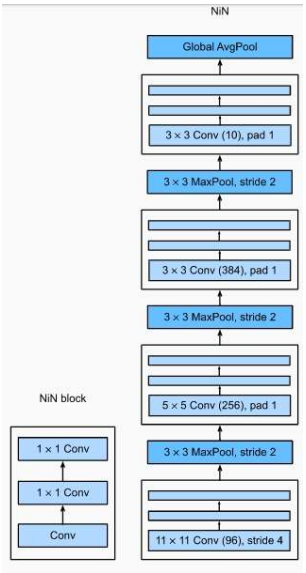
سوال ۲.

به سوالات زیر پاسخ دهید.

الف:

با توجه به بخش ۷.۳ کتاب، مدل NiN را مطالعه نموده و سپس اندازه پارامترهای مدل AlexNet ، VGG و NiN را با GoogLeNet مقایسه کنید.

مدل های LeNet، AlexNet و VGG همگی یک الگوی طراحی مشترک دارند که شامل دنباله ای از لایه های کانولوشنی و pooling است و سپس با چند لایه ی کاملا متصل ادامه پیدا میکنند. مدل های شبکه در شبکه (Network in Network) یا به اختصار NiN بلوک هایی هستند که پیشنهاد شد هر هر بلوک از یک شبکه MLP استفاده شود. به این صورت که برای هر پیکسل یک لایه ی کاملا متصل بزنیم. بلوک های مدل NiN به شکلی طراحی شده اند که تعداد کانال های خروجی بلوک با تعداد کلاس های مسئله برابر باشد و همینطور از اجرای با هم همه ی لایه های کاملا متصل جلوگیری میکند. مدل NiN بشدت تعداد پارامترهای مورد نیاز مدل را کاهش میدهد. شکل زیر معماری NiN را نمایش میدهد.



برای مقایسه تعداد پارامترهای معماری های ذکر شده تعداد پارامترهای مدل برای دیتاست ImageNet را در نظر گرفتیم و بر اساس آن پارامتر ها را مقایسه میکنیم.

برای معماری AlexNet شکل زیر لایه ها و مجموع پارامترهای مدل را نشان میدهد که این مدل حدود ۶۲ میلیون پارامتر را باید برای یادگیری دیتاست ImageNet آموزش دهد.

AlexNet Network - Structural Details													
Input			Output			Layer	Stride	Pad	Kernel size		in	out	# of Param
227	227	3	55	55	96	conv1	4	0	11	11	3	96	34944
55	55	96	27	27	96	maxpool1	2	0	3	3	96	96	0
27	27	96	27	27	256	conv2	1	2	5	5	96	256	614656
27	27	256	13	13	256	maxpool2	2	0	3	3	256	256	0
13	13	256	13	13	384	conv3	1	1	3	3	256	384	885120
13	13	384	13	13	384	conv4	1	1	3	3	384	384	1327488
13	13	384	13	13	256	conv5	1	1	3	3	384	256	884992
13	13	256	6	6	256	maxpool5	2	0	3	3	256	256	0
						fc6			1	1	9216	4096	37752832
						fc7			1	1	4096	4096	16781312
						fc8			1	1	4096	1000	4097000
Total													**62,378,344**

مدل بعدی Vgg است که یکی از مشخصه های معروف بودن این مدل تعداد پارامترهای زیاد آن است که این مدل برای دیتاست ImageNet حدود 138 میلیون پارامتر آموزش میدهد.

VGG16 - Structural Details													
#	Input Image			output			Layer	Stride	Kernel		in	out	Param
1	224	224	3	224	224	64	conv3-64	1	3	3	3	64	1792
2	224	224	64	224	224	64	conv3064	1	3	3	64	64	36928
	224	224	64	112	112	64	maxpool	2	2	2	64	64	0
3	112	112	64	112	112	128	conv3-128	1	3	3	64	128	73856
4	112	112	128	112	112	128	conv3-128	1	3	3	128	128	147584
	112	112	128	56	56	128	maxpool	2	2	2	128	128	65664
5	56	56	128	56	56	256	conv3-256	1	3	3	128	256	295168
6	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
7	56	56	256	56	56	256	conv3-256	1	3	3	256	256	590080
	56	56	256	28	28	256	maxpool	2	2	2	256	256	0
8	28	28	256	28	28	512	conv3-512	1	3	3	256	512	1180160
9	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
10	28	28	512	28	28	512	conv3-512	1	3	3	512	512	2359808
	28	28	512	14	14	512	maxpool	2	2	2	512	512	0
11	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
12	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
13	14	14	512	14	14	512	conv3-512	1	3	3	512	512	2359808
	14	14	512	7	7	512	maxpool	2	2	2	512	512	0
14	1	1	25088	1	1	4096	fc		1	1	25088	4096	102764544
15	1	1	4096	1	1	4096	fc		1	1	4096	4096	16781312
16	1	1	4096	1	1	1000	fc		1	1	4096	1000	4097000
Total													**138,423,208**

GoogleNet با استفاده از ایده ای که داشت باعث شد در کنار افزایش تعداد لایه ها تعداد پارامتر ها کم شود که این مدل حدود 4 میلیون پارامتر دارد.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

و در انتها برای مدل NiN در کنار تعداد لایه های نسبتا زیاد تنها به 2 میلیون پارامتر نیاز داریم که در هزینه محاسباتی بسیار پیشرفت عظیم و خوبی محسوب میشود.

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 54, 54, 96)	34944
conv2d_14 (Conv2D)	(None, 54, 54, 96)	9312
conv2d_15 (Conv2D)	(None, 54, 54, 96)	9312
max_pooling2d_5 (MaxPooling2	(None, 26, 26, 96)	0
conv2d_16 (Conv2D)	(None, 22, 22, 256)	614656
conv2d_17 (Conv2D)	(None, 22, 22, 256)	65792
conv2d_18 (Conv2D)	(None, 22, 22, 256)	65792
max_pooling2d_6 (MaxPooling2	(None, 10, 10, 256)	0
conv2d_19 (Conv2D)	(None, 8, 8, 384)	885120
conv2d_20 (Conv2D)	(None, 8, 8, 384)	147840
conv2d_21 (Conv2D)	(None, 8, 8, 384)	147840
max_pooling2d_7 (MaxPooling2	(None, 3, 3, 384)	0
dropout_2 (Dropout)	(None, 3, 3, 384)	0
conv2d_22 (Conv2D)	(None, 1, 1, 10)	34570
conv2d_23 (Conv2D)	(None, 1, 1, 10)	110
conv2d_24 (Conv2D)	(None, 1, 1, 10)	110
global_average_pooling2d (Gl	(None, 10)	0
dense_3 (Dense)	(None, 1000)	11000

Total params: 2,026,398
Trainable params: 2,026,398
Non-trainable params: 0

ب:

بررسی کنید که چگونه دو معماری شبکه NiN و GoogLeNet به طور قابل توجهی اندازه پارامتر مدل را کاهش می دهند ؟

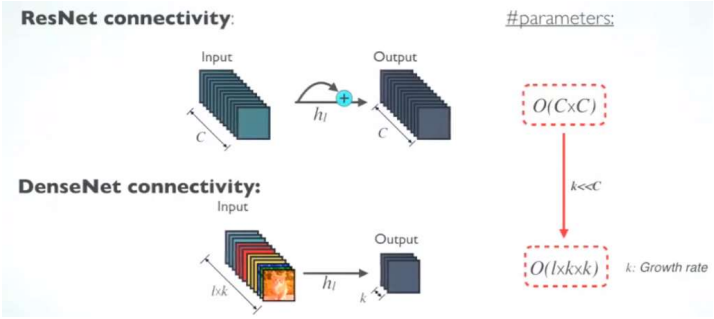
در بلوک های googleNet از تکنیک کانولوشن 1*1 استفاده میشود که باعث کاهش ابعاد نقشه های ویژگی و در نتیجه کاهش چشم گیر تعداد پارامترهای مدل میشود. در اصل تکنیک کانولوشن 1*1 بر خلاف pooling در بعد سوم نقشه های ویژگی کاهش را انجام میدهد این در حالی است که pooling اندازه طول و عرض نقشه ها را کاهش میدهد که باعث میشود مقدار زیادی از اطلاعات در جهت down sampling از بین بروند.

برای مدل NiN همانطورکه در قسمت قبلی سوال نیز توضیح داده شد این مدل با حذف لایه های کاملا متصل و جایگزین کردن آنها با global average pooling هم از overfitting جلوگیری میکند و هم به شدت تعداد پارامترها کاهش میدهد.

ج:

بررسی کنید چرا پارامتر های مدل DenseNet از ResNet کوچکتر است .

شکل زیر دلیل این تفاوت را نمایش میدهد. پارامترهای DenseNet از ResNet کمتر است زیرا در ResNet هر لایه با تعداد C خروجی به لایه ای با C ورودی متصل است(کاملا متصل) که تعداد پارامترها برابر C*C میشود در حالی که در DenseNet اتصالات کاملا متصل نیستند و یک لایه با اندازه l*k به یک لایه با اندازه k متصل میشود که معمولا k نسبت به C بسیار کوچک تر است و تعداد پارامترهای هر لایه در DenseNet که برابر l*k*k میباشد عدد کوچک تری میشود.



سوال ۳.

به سوالات زیر پاسخ دهید .

الف.

کدام یک از مراحل پیش پردازش داده زیر را با نرمال سازی دسته ای و کدام را با نرمال سازی لایه ای میتوان پیاده سازی نمود ؟(با مشخص نمودن مقدار پارامترهای α و β پاسخی که دادید را اثبات نمایید.)

الف – ۱)

کم کردن میانگین تصویر مجموعه داده از هر تصویر در مجموعه داده

میتوان از نرمال سازی دسته ای استفاده کرد. اثبات:

$$\gamma_j = \sigma_j , \beta_j = 0$$

Proof:

$$output\ of\ Batch\ normalization = y_{ij} = \gamma_j \frac{x_{ij} - \mu_{ij}}{\sigma_j} + \beta_j$$

$$y_{ij} = \sigma_j \frac{x_{ij} - \mu_{ij}}{\sigma_j} + 0 = x_{ij} - \mu_{ij}$$

الف – ۲)

مقیاس گذاری هر تصویر از مجموعه داده ها، به طوری که کانال های رنگی برای تمام نقاط یک تصویر مجموعشان یک شود.

میتوان از نرمال سازی لایه ای استفاده کرد. اثبات:

$$\gamma = \frac{\sigma}{N \times D} , \beta = 0$$

Proof:

$$output\ of\ layer\ normalization = y = \gamma \frac{x - \mu}{\sigma} + \beta$$

$$y = \frac{\sigma}{N \times D} \times \frac{x - \mu}{\sigma} + 0 = \frac{x - \mu}{N \times D}$$

ب.

درستی یا نادرستی گزاره های زیر را انتخاب نموده و برای هر کدام به طور مختصر دلیل خود را بنویسید.

ب – ۱)

نرمالسازی دسته ای، پردازش یک دسته را سریع تر میکند و زمان آموزش را کاهش میدهد و در عین حال تعداد به روز رسانی ها را ثابت نگه میدارد. (درست – نادرست)

نادرست – گزاره بالا شامل سه مورد است. مورد اول در مورد پردازش یک دسته سرعت کاهش میابد به دلیل انجام محاسبات بیشتری که باید صرف شود. قسمت دوم در مورد شبکه درست است زیرا اگر از batch normalization استفاده کنیم شبکه با سرعت بیشتری همگرا میشود و در مورد تعداد به روز رسانی ها نیز گزاره نادرست است زیرا تعداد به روز رسانی ها با نرمالسازی دسته ای متغیر است و ثابت نمی ماند.

ب – ۲)

نرمالسازی دسته ای اتصال بین لایه های قبلی و بعدی را ضعیف میکند، که سبب یادگیری مستقل در شبکه میشود.(درست – نادرست)

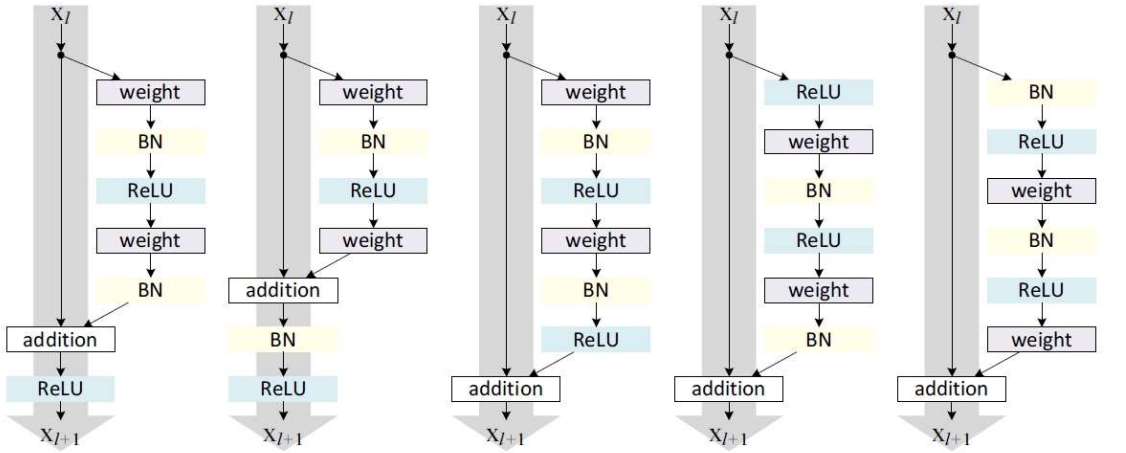
درست – Batch normalization باعث میشود شبکه حساسیت کمتری به مقدار دهی اولیه وزن ها داشته باشد. زیرا در شبکه های عمیق در حالتی که از normalization استفاده نکنیم مقادیر هر نرون به نرون های قبلی خود وابسته است پس شبکه مجبور میشود با تغییرات در توزیع ورودی منطبق شود و در این حالت تغییرات در لایه های ابتدایی در لایه های ابتدایی باعث تغییرات بسیار زیادی درلایه های پایانی میشود. اما استفاده از نرمال سازی از این اتفاق جلوگیری میکند.

سوال ۴.

در نسخه های نهایی ResNet ، ساختار "لایه همگشتی، نرمالسازی دسته ای و تابع فعالسازی" به ساختار"نرمالسازی دسته ای، تابع فعالسازی و لایه همگشتی" تغییر یافته است و معماری DenseNet نیز از این ساختار پیروی میکند. بنظر شما این تغییر چه مزیتی دارد ؟

در رفرنس [8] مراجع این تمرین نویسندگان مقاله با تغییر ترتیب residual block اصلی، ترکیب نرمالسازی دسته ای، تابع فعالسازی و کانولوشن را پیشنهاد دادند و که در آن گرادیان میتواند از اتصالات کوتاه به هر لایه ی قبلی، بدون مانع منتقل شود. در معماری اصلی residual block بعد از بالا بردن تعداد لایه ها، گرادیان برای انتقال به لایه های قبلی دچار میشکل میشود. نویسندگان رفرنس [8] ادعا میکنند که با تغییر این ساختار توانایی آموزش یک شبکه ResNet با عمق 1001 لایه را نیز دارند. شکل زیر معماری های مختلف residual block ها در مقالات مختلف به همراه جدول مقایسه ی عملکرد (خطای طبقه بندی) روی دیتاست CIFAR10 را نمایش میدهد.

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**



References

1) https://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception

2) medium.com/analytics-vidhya/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more

3) https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/

4) https://medium.com/mlearning-ai/an-overview-of-vgg16-and-nin-models

5) towardsdatascience.com/densenet-image-classification- Number_of_Parameters_for_ResNet

6) https://github.com/cvjena/cnn-models/issues/3

7) https://www.quora.com/What-is-the-order-of-using-batch-normalization-Is-it-before-or-after-activation-function

8) K. He, X. Zhang, S. Ren, and J. Sun. Identity Mappings in Deep Residual Networks. arXiv preprint arXiv:1603.05027v3,2016.

9) https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035

10) https://github.com/KaimingHe/resnet-1k-layers

11) https://deeplearning.ir/%D9%85%D8%B9%D8%B1%D9%81%DB%8C-batchnormalization/

12) towardsdatascience.com/Batchnormalizationoptimizesnetworktrainingduring back propagation.